

FIG. 2 is a flow diagram showing the steps of a prior art method for accessing a class that is needed;

FIG. 3 is a flow diagram of a prior art method for a class loader to perform the step of loading a class in step 220 of FIG. 2;

FIG. 4 is a block diagram of an apparatus in accordance with the preferred embodiments;

FIG. 5 is a flow diagram showing the steps of a method in accordance with the preferred embodiments for accessing a class that is needed;

FIG. 6 is a flow diagram of a method in accordance with the preferred embodiments for a class loader to perform the step of loading a class in step [530] 520 of FIG. 5;

FIG. 7 is a flow diagram of one suitable method for performing check 1 in step 632 of FIG. 6;

FIG. 8 is a flow diagram of one suitable method for performing check 2 in step 652 of FIG. 6;

FIG. 9 is a flow diagram of one suitable method for performing check 3 in step 662 of FIG. 6;

FIG. 10 is a flow diagram of one suitable method for performing the runtime checks in step [550] 540 of FIG. 5;

FIG. 11 is a list of Java-defined reflection methods that allow accessing dynamically-defined classes;

FIG. 12 is a list of Java Native Interface (JNI) functions that can access Java classes from outside the Java program;

FIG. 13 is a prior art JVM build process; and

FIG. 14 is a JVM build process for system level code in accordance with the preferred embodiments.

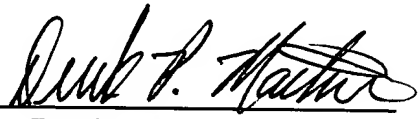
Please delete the paragraph at p. 16 line 17 to p. 17 line 3, and substitute therefor:

Referring now to FIG. 5, a method 500 shows the steps to load and use system level Java classes in accordance with the preferred embodiments. Note that method 500 is similar in some ways to method 200 of FIG. 2, but includes a call to a new class loader in step [530] 520, and performs runtime checks before using a class in step [550] 540. Method 500 is performed when a class is needed. If the JVM already has an internal representation of the needed class (step 510=YES), the JVM does not need to load the class, so the loading of the class is done (step 530). If, however, the JVM does not have an internal representation of the class (step 510=NO), method 500 gets the needed class from a new class loader (step 520), the details of which are described below with reference to FIG. 6. Once method 500 is done loading the class (step 530), method 500 performs runtime checks before using the class (step 540) to assure that only authorized code can call the protected Java classes.



The amendments to the specification are the only portion of the amendment filed on 12/10/03 that were non-compliant, so this is the only portion that is included herein. Applicants therefore rely on the remainder of the amendment filed on 12/10/03 in response to the office action dated 09/11/03. The Examiner is invited to telephone the undersigned if this would in any way advance the prosecution of this case.

Respectfully submitted,

By 

Derek P. Martin
Reg. No. 36,595

MARTIN & ASSOCIATES, L.L.C.
P.O. Box 548
Carthage, MO 64836-0548
(417) 358-4700

RECEIVED

JAN 14 2004

Technology Center 2100